Accelerating FUN3D v13.7 Solutions Using GPU Hardware

Aaron Walden, Eric Nielsen, and Gabriel Nastac
Computational AeroSciences Branch
NASA Langley Research Center

March 2, 2021

Enduring Partnership with NVIDIA: Thanks!

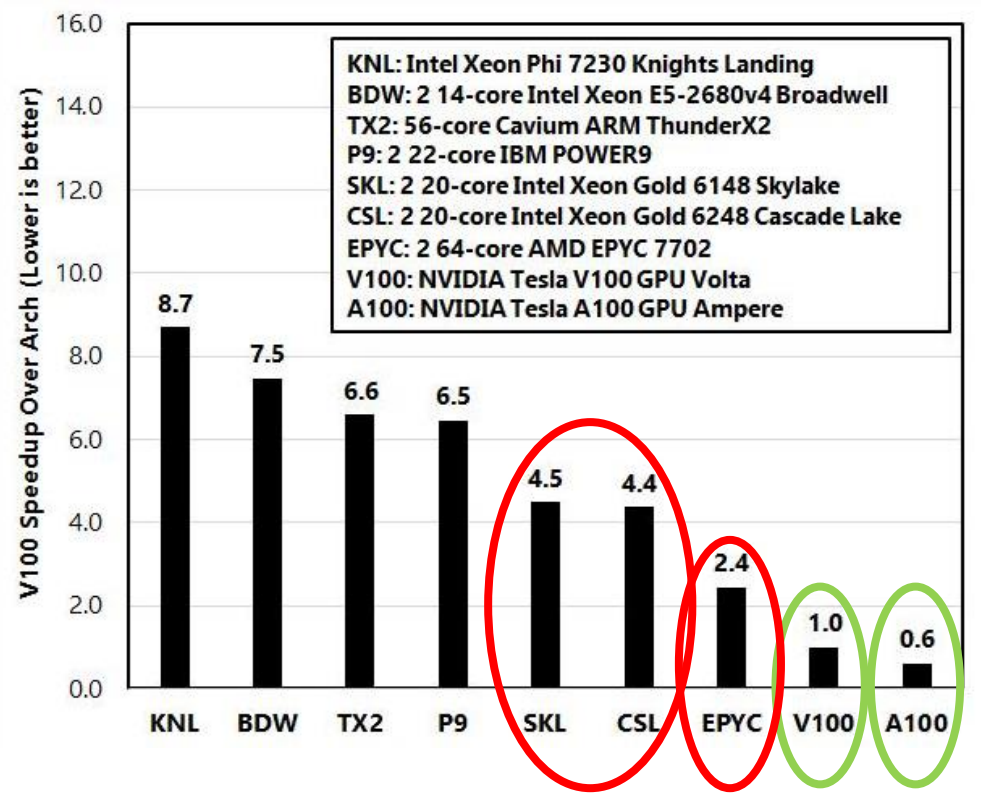
Teammate Ashley Korzun onstage with NVIDIA CEO Jensen Huang at Supercomputing 2019

Disclaimers

- This session assumes you have basic working knowledge of running FUN3D on CPUs
- In addition to the material included here, please be sure to read the GPU chapter of the user manual and corresponding section describing the `&gpu_support` namelist
  - There are ***many*** more details and tips/tricks covered there

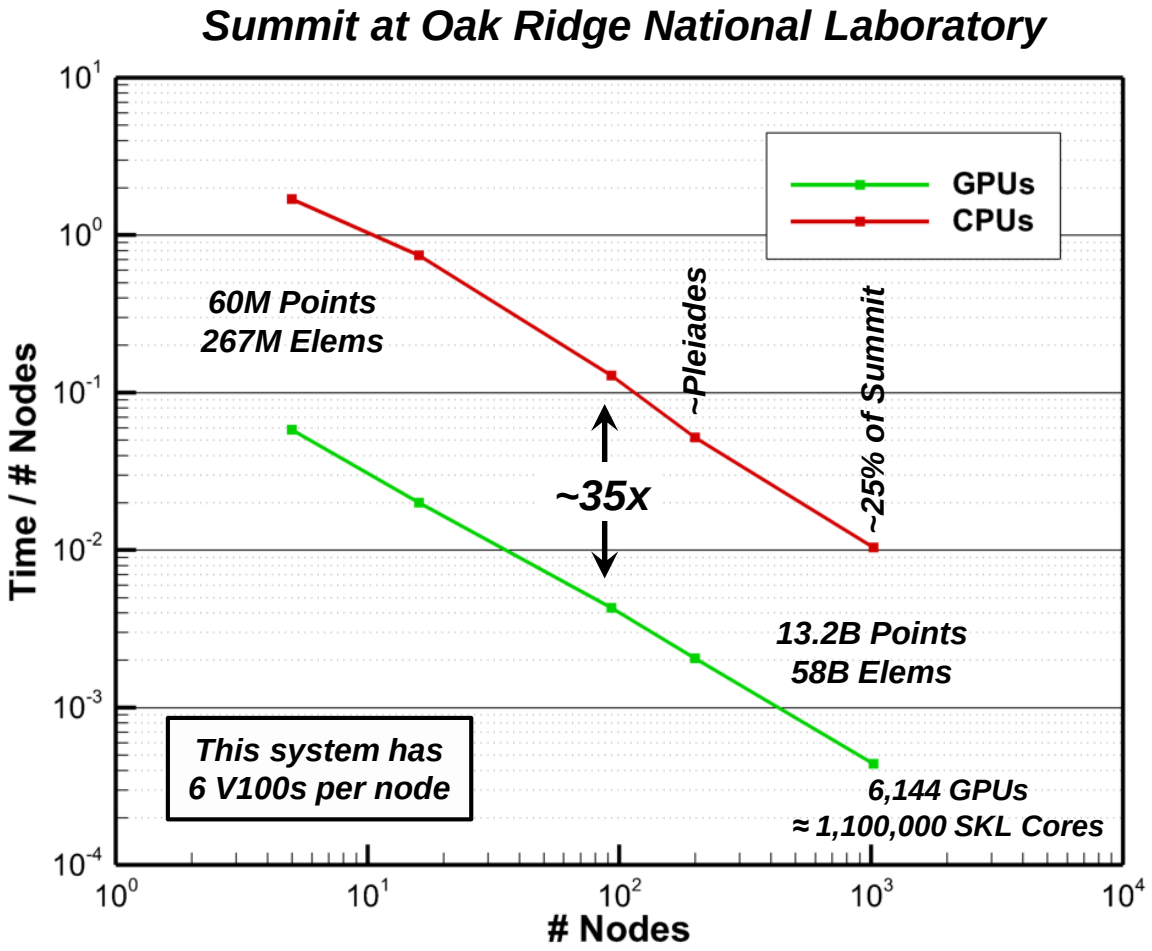
Some Background

- FUN3D team has explored the use of GPUs for about 10 years: the software/hardware ecosystem has advanced quickly
- Breakthrough performance achieved in 2017 after adopting CUDA and the introduction of the NVIDIA Tesla V100
- Have spent the past two years hardening the implementation and beta testing with early adopters inside and outside NASA
- Like many large-scale science codes, FUN3D is memory bound, which means performance is primarily limited by the rate at which data can be moved from main memory to the processing units
  - The memory bandwidth offered by the V100 is 900 GB/s, substantially higher than the ~250 GB/s available on a typical Intel Xeon
- A single V100 will provide the performance of 180-200 Xeon Skylake cores, depending on FUN3D input options
- For perfect gas RANS, we suggest at least 1 million grid points (not elements) on each GPU
  - If you are not seeing ~0.15 seconds per step, per 1 million grid points on V100, things are not working correctly
  - Can go less than 1 million points per GPU, but you will not be using the GPU effectively
  - Can fit ~7 million points in 32 GB of GPU memory



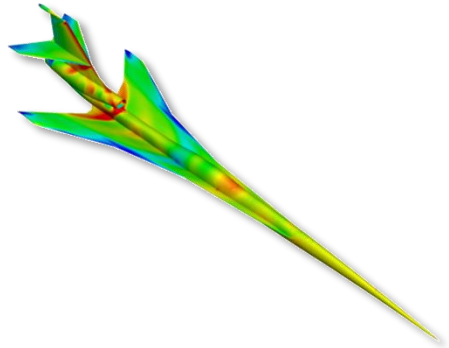
Typical Hardware Setups

- GPUs still require a CPU to run the OS and other applications
  - The GPU option in FUN3D executes all kernels required for time stepping on the GPU, avoiding expensive data motion b/w *host* and *device*
  - Some FUN3D features remain on the host (e.g., preprocessing, visualization options); we will discuss how to mitigate these costs
- Most GPU-based systems now come with 4, 6, or 8 GPUs per node
  - Node-level performance is considerably faster than what you are used to: A node of 4 V100s will perform similar to 700-800 Xeon Skylake cores
  - Such nodes are more expensive, but performance/dollar is still a win

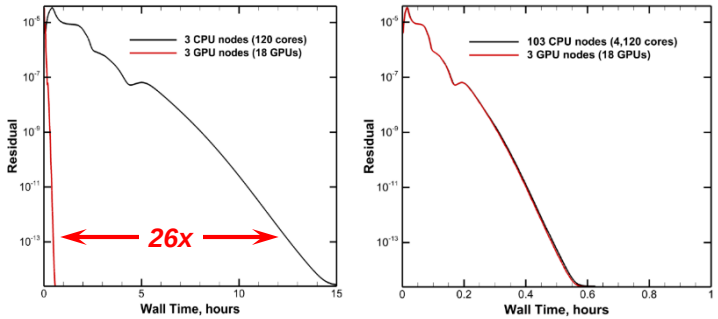


Performance for Capacity Jobs

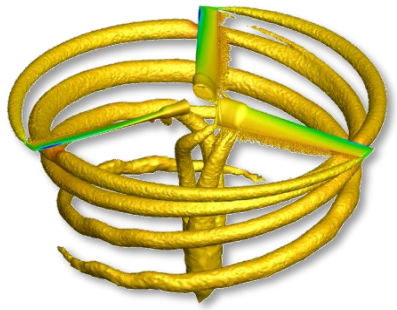
- NVIDIA Tesla V100 GPU outperforms Intel Xeon Skylake CPU by 4-5x
  - New NVIDIA Tesla A100 GPU improves to 7-8x
- GPUs typically bundled in nodes with 4, 6, or 8 GPUs
- GPU nodes are more expensive, but still a win on performance / $



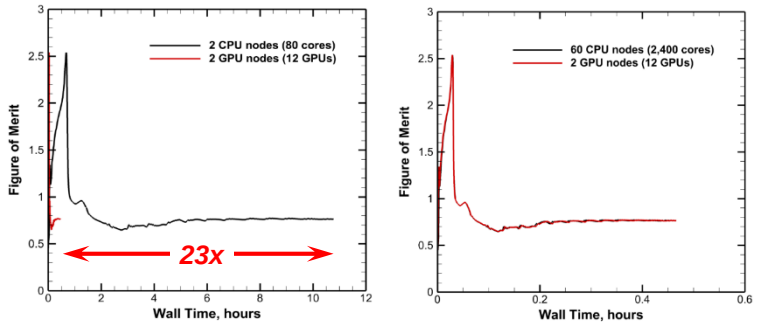
Supersonic Flows



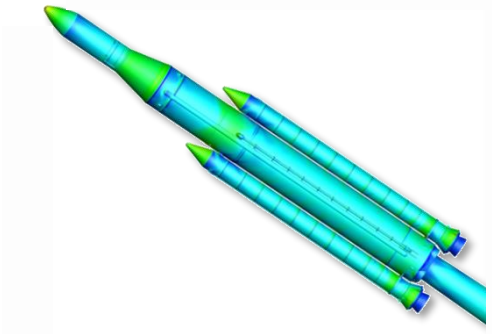
3 GPU nodes of 6xV100: 37 mins
3 CPU nodes of 120 cores: 16 hrs
OR
18 GPUs do the work of 103 CPUs (4,120 cores)



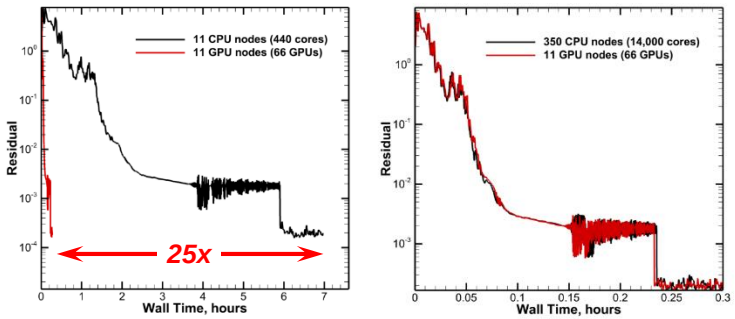
Rotorcraft



2 GPU nodes of 6xV100: 28 mins
2 CPU nodes of 80 cores: 11 hrs
OR
12 GPUs do the work of 60 CPUs (2,400 cores)



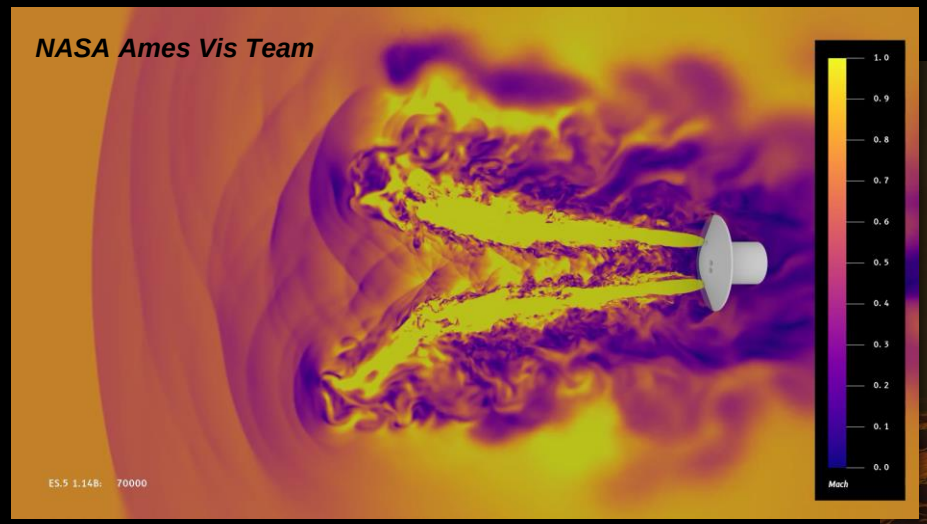
Space Launch System

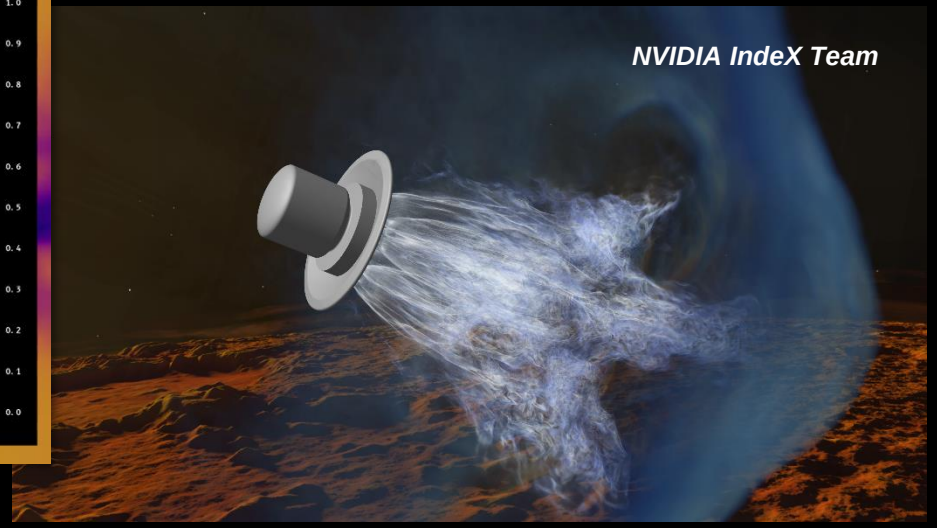


11 GPU nodes of 6xV100: 17 mins
11 CPU nodes of 440 cores: 7 hrs
OR
66 GPUs do the work of 350 CPUs (14,000 cores)

Performance for Capability Jobs

- Simulated ensembles of human-scale Mars landers investigating effects of throttle settings, flight Mach number, and spatial resolutions to billions of grid points
- Simulations of unprecedented spatial and temporal fidelity
- 40 TB/day migrated from ORNL to NASA Ames
- Largest public domain unstructured-grid CFD datasets available to the international visualization community, *https://data.nas.nasa.gov/fun3d/*
- Game-changing computational performance
  - Xeon: One run in ~9 months on 5,000 SKL cores with 10-day waits for 5-day jobs
  - Summit: Six runs done in 4.5 days on 3,312 GPUs





System Requirements

General

- At present, FUN3D will only execute on NVIDIA Tesla GPUs; use at least the P100 (Pascal), but prefer V100 (Volta) or A100 (Ampere). Do not use Kepler or earlier.
- Host may be an Intel or AMD x86 CPU, IBM POWER CPU, or a compatible ARM CPU
- Version 10.2 or higher of the NVIDIA CUDA Toolkit must be installed

MPI Concerns

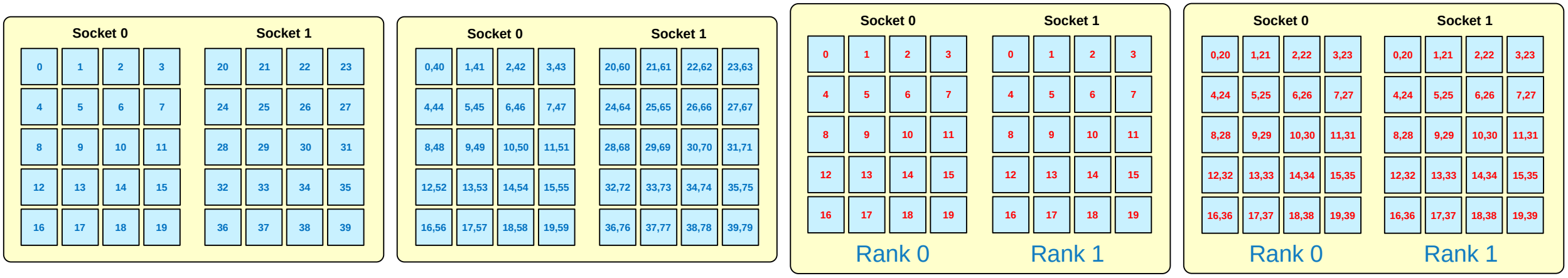
- An NVIDIA NVLink interconnect may provide improved performance, but is not necessary
- Execution across multiple GPUs has been tested with the following MPI implementations:
  - OpenMPI, Intel MPI, HPE/SGI MPT, IBM Spectrum MPI, MVAPICH, Cray MPI
- By default, FUN3D will automatically determine how to assign MPI ranks to GPUs
  - But many variations here; see the user manual for details or email fun3d-support@lists.nasa.gov
- FUN3D can utilize CUDA-enabled MPI; however, performance is very sensitive to the communication stack installed on the system – see the FUN3D user manual or contact fun3d-support@lists.nasa.gov for help. Host-based MPI calls are generally more efficient.

Node-Level Partitioning, MPI, and Threads

- For FUN3D, you can consider grid partitions and MPI ranks as synonymous
- During vanilla CPU-based execution of FUN3D:
  - In a flat MPI setting, one MPI rank is generally assigned to each CPU core (Case A)
    - Multiple MPI ranks may be assigned to each CPU core if multithreading (Case B)
  - In a hybrid MPI-OpenMP setting, a single MPI rank may be assigned to a NUMA domain (socket) and OpenMP threads may be spawned at the loop level to populate the cores (Cases C, D)

Consider a dual-socket Xeon Skylake with 20 cores/socket:

MPI Rank **OpenMP Thread**



Case A
40 MPI Ranks

Case B
80 MPI Ranks

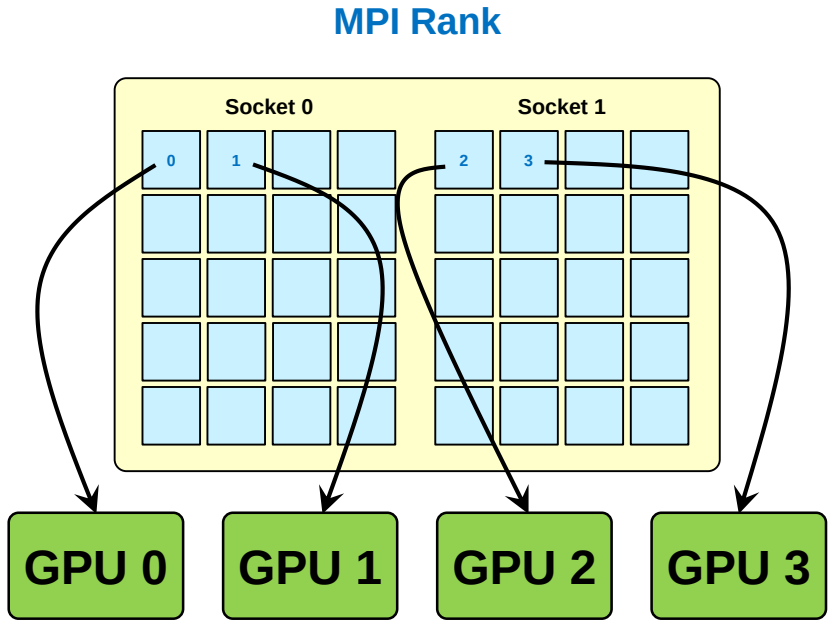
Case C
2 MPI Ranks, 1 Thread/Core

Case D
2 MPI Ranks, 2 Threads/Core

Node-Level Partitioning, MPI, and GPUs

- When using GPUs to accelerate FUN3D, the simplest and most efficient strategy is to assign a single MPI rank to each GPU, with the ranks spread out evenly over the sockets
  - Recall that no time-stepping kernels execute on the host when FUN3D is using GPU acceleration, so these CPU cores are solely used to launch CUDA kernels: they serve only to direct traffic
  - Note that most CPU cores sit idle in this paradigm: if host-based kernels such as preprocessing and visualization support are major contributors to the workflow, this arrangement may yield poor performance for such kernels

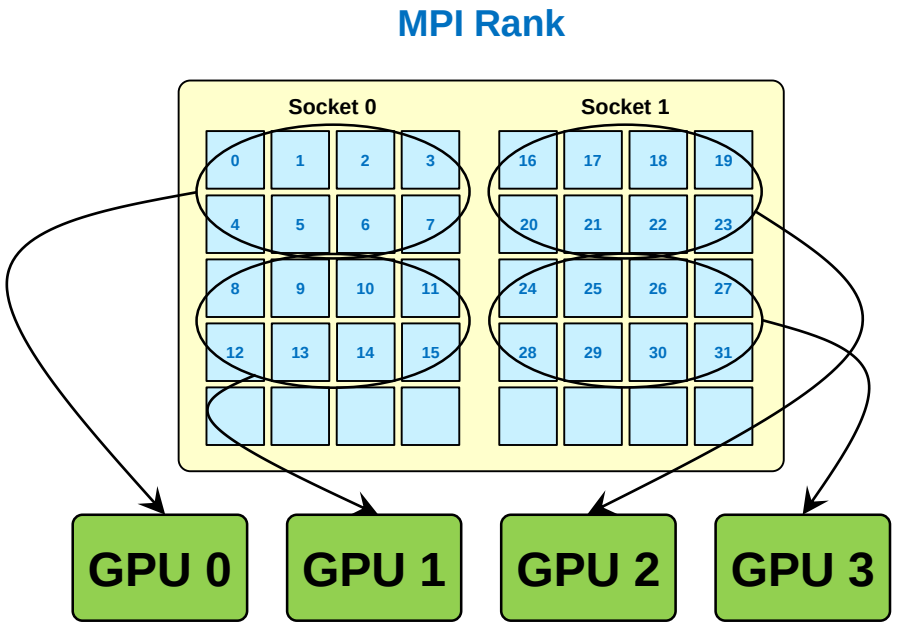
Consider a dual-socket Xeon Skylake with 20 cores/socket and a 4xV100 arrangement:



Running with Multiple MPI Ranks per GPU

- If host-based kernels such as preprocessing and visualization support are critical to overall performance, we can instantiate more MPI ranks to distribute over the CPU cores
- In this case, launch the MPI job with an integer multiple of the number of GPUs present, and FUN3D will assign multiple MPI ranks to each GPU
- FUN3D will require the use of NVIDIA's Multi-Process Service (MPS), which allows multiple CUDA kernels to be processed simultaneously on the same GPU
- Host-based kernels will now scale accordingly, while GPU performance should not degrade more than 5-10% when using up to 8 MPI ranks per GPU

Consider a dual-socket Xeon Skylake with 20 cores/socket and a 4xV100 arrangement:



NAS-Specific Guidance

- NAS provides 48 nodes of 4xV100 GPUs (and two 8xV100 nodes)
- For this session, we will use the following prebuilt v13.7 module:

```
module use --append /swbuild/fun3d/shared/fun3d_users/modulefiles
module load FUN3D_AVX512/13.7
```

- Here we assume use of entire nodes of 4 GPUs; see online NAS documentation for requesting partial nodes
- GPU jobs should be submitted to the **v100** queue; it is accessed via the PBS server pbspl4 using one of the following methods:
  - Use `#PBS -q v100@pbspl4` in your PBS script
  - Use `-q v100@pbspl4` in your qsub command
  - Log into pbspl4 and submit your job there
- Unlike most other GPU systems, your script must contain the line

```
unset CUDA_VISIBLE_DEVICES    # bash
unsetenv CUDA_VISIBLE_DEVICES # csh
```

- For current guidance on running FUN3D on NAS GPUs, enter the command:

```
module help /path/to/your/FUN3D/module
```

- For more details, see the online NAS documentation

General Setup Guidance

- Build your copy of FUN3D with GPU support
  - See Appendix A of User Manual; configure with:

    `--with-cuda=/path/to/CUDA` Path to your CUDA installation

    `--with-libfluda=/path/to/FLUDA` Path to your fluda_binaries/x86_64/single_precision in the v13.7 tarball
- Here we assume use of entire nodes of 4 GPUs
- See your system documentation for guidance on how to submit jobs to GPU-enabled resources
- Please contact fun3d-support@lists.nasa.gov for assistance; we are happy to help

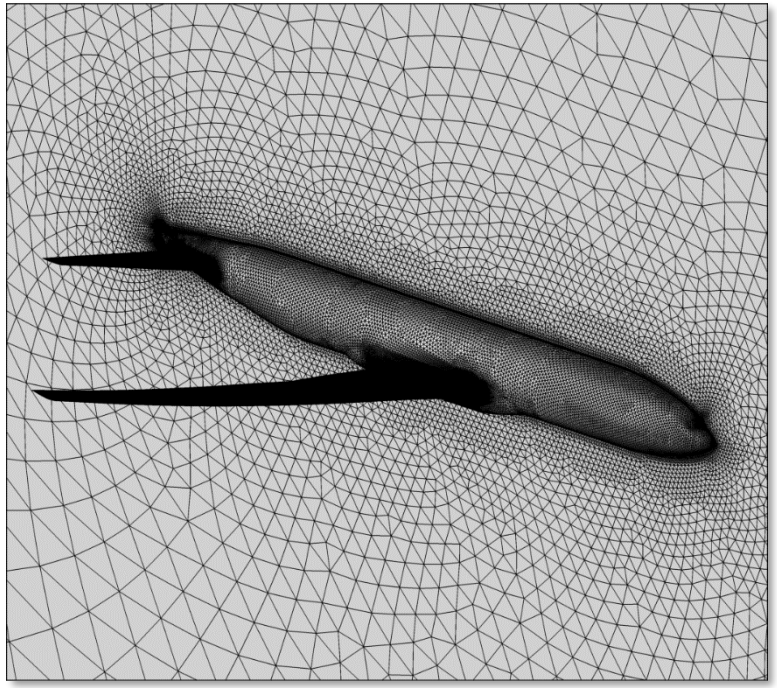
Supported FUN3D Options

- Always see the current FUN3D manual for a complete list
- All runs on tetrahedral grids must set `mixed=.true.` in the `&code_run_control` namelist
- All visualization options are available, but will execute on the CPU (boundaries, volume, slices, sampling, etc). More details later in the session.
- All of the usual output files will be produced ([project].forces, [project].flow, [project]_hist.dat…)

- Fluxes: Roe, LDFSS, Low-dissipation Roe
- Limiters: Barth, h-van Albada (both may be frozen if desired)
- Jacobians: van Leer, LDFSS
- Turbulence modeling: Legacy Spalart-Allmaras model with DES, DDES, and MDDES options; Dacles-Mariani, RC options
- QCR2000
- Boundary conditions: 3000, 4000 (optional adiabatic wall, specified surface velocities), 5000, 5050, 5051, 5052, 6662, 7011, 7012, 7100, 7201
- Time integration: Backward Euler for steady flows, all unsteady BDF schemes, local time-stepping
- Time-averaging statistics
- Specified rigid grid motion
- Aeroelastic analysis using internal modal solver with modal mesh deformation
- Asynchronous native volume and surface pressure outputs

Demo Cases

- Three grids chosen from the 4th AIAA Drag Prediction workshop, based solely on grid size
- Original grids were tetrahedral; merged into mixed elements for this exercise
- Mach 0.85, Re=5M, AOA=1 deg; Spalart-Allmaras turbulence model
- Each case is run for 500 time steps



	Grid 1: “1M”	Grid 2: “6M”	Grid 3: “10M”
Points	1,233,948	5,937,410	10,252,669
Tetrahedra	983,281	7,815,201	14,836,294
Pyramids	22,866	71,789	89,642
Prisms	2,068,172	9,006,159	15,154,594

Running on a Single GPU

- Get the grid and fun3d.nml: `wget https://fun3d.larc.nasa.gov/GPUShortCourse/1M.tgz`
- Here, we are using a single CPU core as a shepherd for a single GPU; **all other CPU cores sit idle**

fun3d.nml

```
&project
  project_rootname = 'dpw-wb0_med-7Mc_5.merged'
/
&raw_grid
  grid_format = 'aflr3'
  data_format = 'stream'
/
&reference_physical_properties
  angle_of_attack    = 1.0
  mach_number        = 0.85
  reynolds_number    = 18129.1
  temperature        = 560.0
  temperature_units = 'Rankine'
/
&force_moment_integ_properties
  area_reference  = 594720.0
/
&nonlinear_solver_parameters
  schedule_cfl     = 10.0 200.0
  schedule_cflturb =  1.0  30.0
/
&code_run_control
  steps               = 500
  restart_read        = 'off'
/
&gpu_support
  use_cuda = .true.
/
```

PBS Script

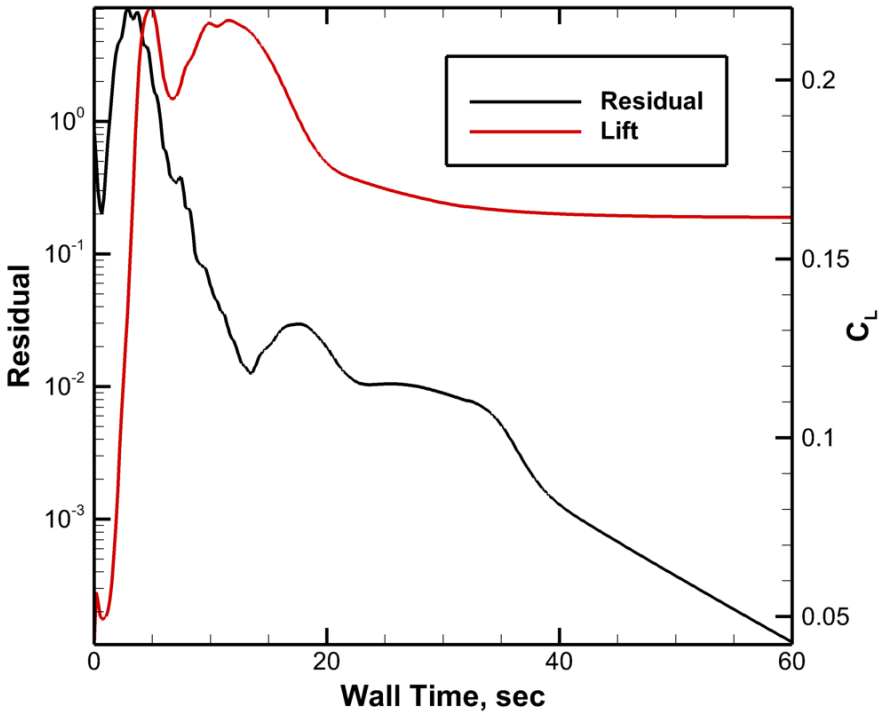
```
#PBS -S /bin/csh
#PBS -N run_test
#PBS -r n
#PBS -m ae
#PBS -M eric.j.nielsen@nasa.gov
#PBS -l select=1:ncpus=36:mpiprocs=36:model=sky_gpu:ngpus=4
#PBS -l walltime=0:10:00
#PBS -q v100@pbspl4

module use --append /swbuild/fun3d/shared/fun3d_users/modulefiles  # NASA ONLY
module purge                                                       # NASA ONLY
module load FUN3D_AVX512/13.7                                      # NASA ONLY

unsetenv CUDA_VISIBLE_DEVICES                                      # NASA ONLY

((mpiexec_mpt -np 1 nodet_mpi --time_timestep_loop ) > test.out) >& error.out
```

Running on a Single GPU

Screen Output

```
FUN3D 13.6-0b87d96d13 Flow started 10/11/2020 at 11:52:40 with 1 processes
Contents of fun3d.nml file below-----------------------
&project
  project_rootname = 'dpw-wb0_med-7Mc_5.merged'
/
.
.
.
WARNING: CUDA MPS NOT running on r101i0n10.
CUDA MPS status is good: either not needed or running properly on all 1 nodes.
.
.
.
   52  0.216158829571317E+00  0.46052E+02  0.40292E+04  0.41325E+04  0.57527E+04
       0.142220594351411E+01  0.25588E+03  0.98227E+04  0.31565E+04  0.13871E+04
       Lift  0.204387766282627E+00          Drag  0.171265575314454E-01
.16329447 seconds to complete timestep on the master rank.
   53  0.221866818685588E+00  0.47732E+02  0.40292E+04  0.41325E+04  0.57527E+04
       0.146033086295375E+01  0.25335E+03  0.98227E+04  0.31565E+04  0.13871E+04
       Lift  0.205038956434613E+00          Drag  0.168285531266851E-01
.10379885 seconds to complete timestep on the master rank.
   54  0.208703702628507E+00  0.49355E+02  0.72370E+04  0.51382E+04 -0.26047E+04
       0.150686999090736E+01  0.25100E+03  0.98227E+04  0.31565E+04  0.13871E+04
       Lift  0.205646859090078E+00          Drag  0.165874305015577E-01
.16937434 seconds to complete timestep on the master rank.
.
.
.
61.951 seconds to complete main timestep loop on the master rank.
 Done.
```

- Running with a single MPI rank
- MPS is not running (and is not needed; more later)
- Nominal time step costs 0.16 seconds
  - As we converge, Jacobian evaluations are frequently skipped, reducing per-step costs to 0.10 seconds



Running on a Single Node with 4 GPUs

- Get the grid and fun3d.nml: `wget https://fun3d.larc.nasa.gov/GPUShortCourse/6M.tgz`
- Here, we are using four CPU cores as shepherds for four GPUs; **all other CPU cores sit idle**

fun3d.nml

```
&project
  project_rootname = 'dpw_wbt0_fine-35Mc_5.merged'
/
&raw_grid
  grid_format = 'aflr3'
  data_format = 'stream'
/
&reference_physical_properties
  angle_of_attack    = 1.0
  mach_number        = 0.85
  reynolds_number    = 18129.1
  temperature        = 560.0
  temperature_units = 'Rankine'
/
&force_moment_integ_properties
  area_reference  = 594720.0
/
&nonlinear_solver_parameters
  schedule_cfl     = 10.0 200.0
  schedule_cflturb =  1.0  30.0
/
&code_run_control
  steps              = 500
  restart_read       = 'off'
/
&gpu_support
  use_cuda = .true.
/
```

PBS Script

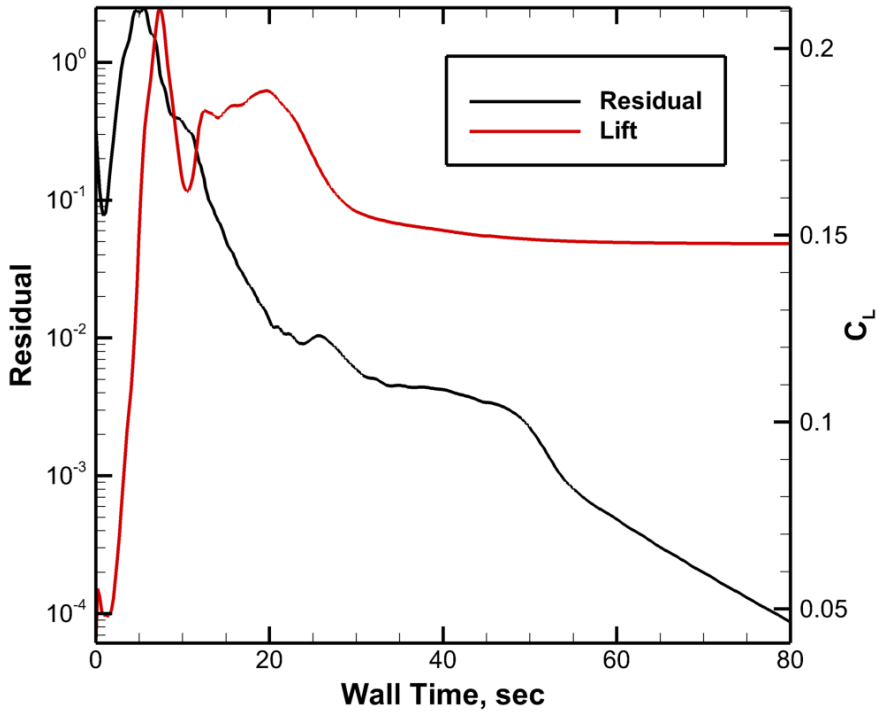
```
#PBS -S /bin/csh
#PBS -N run_test
#PBS -r n
#PBS -m ae
#PBS -M eric.j.nielsen@nasa.gov
#PBS -l select=1:ncpus=36:mpiprocs=36:model=sky_gpu:ngpus=4
#PBS -l walltime=0:10:00
#PBS -q v100@pbspl4

module use --append /swbuild/fun3d/shared/fun3d_users/modulefiles # NASA ONLY
module purge                                                      # NASA ONLY
module load FUN3D_AVX512/13.7                                     # NASA ONLY

unsetenv CUDA_VISIBLE_DEVICES                                     # NASA ONLY

((mpiexec_mpt -np 4 nodet_mpi --time_timestep_loop ) > test.out) >& error.out
```

Running on a Single Node with 4 GPUs

Screen Output

```
FUN3D 13.6-0b87d96d13 Flow started 10/11/2020 at 11:56:22 with 4 processes
Contents of fun3d.nml file below-----------------------
&project
  project_rootname = 'dpw_wbt0_fine-35Mc_5.merged'
/
.
.
.
WARNING: CUDA MPS NOT running on r101i0n10.
CUDA MPS status is good: either not needed or running properly on all 1 nodes.
.
.
.
   62  0.951085003205109E-01  0.43358E+02  0.10865E+05  0.20825E+04  0.87569E+03
       0.856370589209556E+00  0.33632E+03  0.10865E+05  0.20825E+04  0.87569E+03
       Lift  0.182665507664676E+00          Drag  0.150035784519962E-01
.21863349 seconds to complete timestep on the master rank.
   63  0.905743990845053E-01  0.41046E+02  0.10795E+05  0.41044E+04  0.34631E+04
       0.852604223912869E+00  0.32636E+03  0.10865E+05  0.20825E+04  0.87569E+03
       Lift  0.182568468979598E+00          Drag  0.151284763517011E-01
.14476674 seconds to complete timestep on the master rank.
   64  0.851303952100990E-01  0.39652E+02  0.10795E+05  0.41044E+04  0.34631E+04
       0.847837283747530E+00  0.31391E+03  0.10865E+05  0.20825E+04  0.87569E+03
       Lift  0.182203611038651E+00          Drag  0.152642508523030E-01
.21870519 seconds to complete timestep on the master rank.
.
.
.
87.605 seconds to complete main timestep loop on the master rank.
 Done.
```

- Running with four MPI ranks
- MPS is not running (and is not needed; more later)
- Nominal time step costs 0.22 seconds
  - As we converge, Jacobian evaluations are frequently skipped, reducing per-step costs to 0.14 seconds



Running on Two Nodes with 4 GPUs Each

- Get the grid and fun3d.nml: `wget https://fun3d.larc.nasa.gov/GPUShortCourse/10M.tgz`
- Here, we are using four CPU cores as shepherds for four GPUs on each of two nodes; **all other CPU cores sit idle**

fun3d.nml

```
&project
  project_rootname = 'dpw_wbt0_fine-35Mc_5.merged'
/
&raw_grid
  grid_format = 'aflr3'
  data_format = 'stream'
/
&reference_physical_properties
  angle_of_attack    = 1.0
  mach_number        = 0.85
  reynolds_number    = 18129.1
  temperature        = 560.0
  temperature_units = 'Rankine'
/
&force_moment_integ_properties
  area_reference  = 594720.0
/
&nonlinear_solver_parameters
  schedule_cfl     = 10.0 200.0
  schedule_cflturb =  1.0  30.0
/
&code_run_control
  steps               = 500
  restart_read        = 'off'
/
&gpu_support
  use_cuda = .true.
/
```

PBS Script

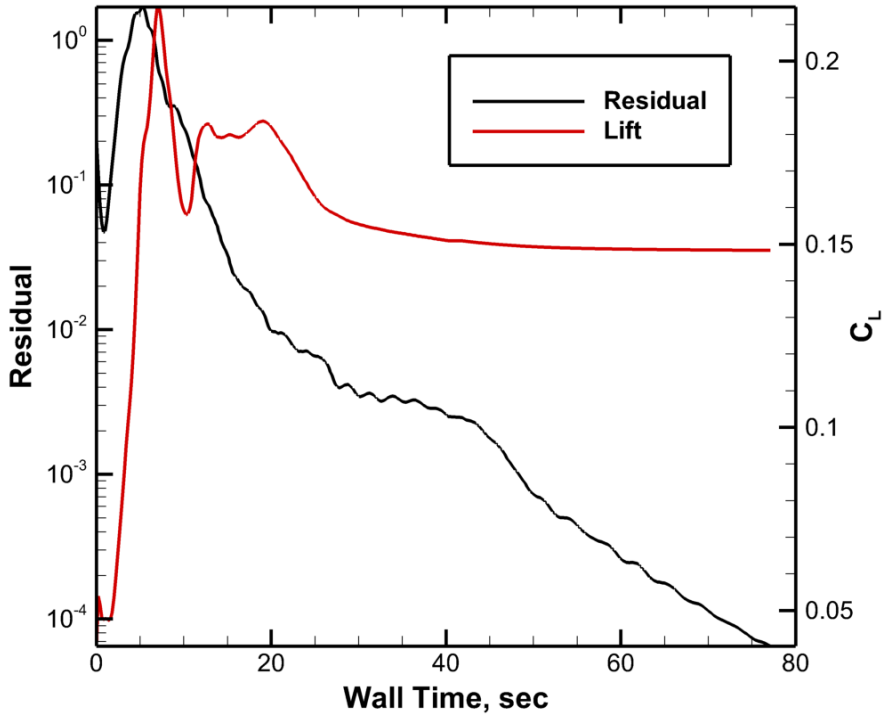
```
#PBS -S /bin/csh
#PBS -N run_test
#PBS -r n
#PBS -m ae
#PBS -M eric.j.nielsen@nasa.gov

#PBS -l select=2:ncpus=36:mpiprocs=4:model=sky_gpu:ngpus=4:mem=300g
#PBS -l place=scatter:excl
#PBS -l walltime=0:10:00
#PBS -q v100@pbspl4


module use --append /swbuild/fun3d/shared/fun3d_users/modulefiles # NASA ONLY
module purge                                                      # NASA ONLY
module load FUN3D_AVX512/13.7                                     # NASA ONLY

unsetenv CUDA_VISIBLE_DEVICES                                     # NASA ONLY

((mpiexec_mpt -np 8 nodet_mpi --time_timestep_loop ) > test.out) >& error.out
```

Running on Two Nodes with 4 GPUs Each

Screen Output

```
FUN3D 13.6-237971ad24 Flow started 10/12/2020 at 07:56:12 with 8 processes
Contents of fun3d.nml file below-----------------------
&project
  project_rootname = 'dpw-wbt0_med-10Mn_5.merged'
/
.
.
.
WARNING: CUDA MPS NOT running on r101i0n10.
WARNING: CUDA MPS NOT running on r101i0n11.
CUDA MPS status is good: either not needed or running properly on all 2 nodes.
.
.
.
   60  0.132018484698159E+00  0.14594E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       0.655829947965633E+00  0.51355E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.176613064418888E+00          Drag  0.153978023947141E-01
.20061884 seconds to complete timestep on the master rank.
   61  0.120967899839322E+00  0.11989E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       0.633808927154000E+00  0.47335E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.178788035714884E+00          Drag  0.148029028171359E-01
.13077858 seconds to complete timestep on the master rank.
   62  0.108924998160599E+00  0.88673E+02  0.16541E+05  0.00000E+00 -0.53707E+02
       0.628431902834406E+00  0.45012E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.180312537215482E+00          Drag  0.143854284486610E-01
.19415972 seconds to complete timestep on the master rank.
.
.
.
79.732 seconds to complete main timestep loop on the master rank.
 Done.
```

- Running with eight MPI ranks
- MPS is not running (and is not needed; more later)
- Nominal time step costs 0.20 seconds
  - As we converge, Jacobian evaluations are frequently skipped, reducing per-step costs to 0.13 seconds



Running Multiple MPI Ranks per GPU

- Recall we have only used a very small number of MPI ranks per CPU so far
  - This severely hampers the performance of CPU kernels such as preprocessing and visualization
- To mitigate these bottlenecks, we may run a larger number of MPI ranks, with multiple ranks sharing a GPU
  - Choose an integer multiple of the number of GPUs available
- To facilitate efficient sharing of each GPU, use the NVIDIA Multi-Process Service (MPS)
  - You may start this daemon yourself, or have FUN3D do it internally
- Here, we are using 32 CPU cores as shepherds for four GPUs (8 MPI ranks each) on each of two nodes; **all other CPU cores sit idle**

fun3d.nml

```
.
.
.
&gpu_support
  use_cuda        = .true.
  cuda_start_mps  = .true.
/
```

PBS Script

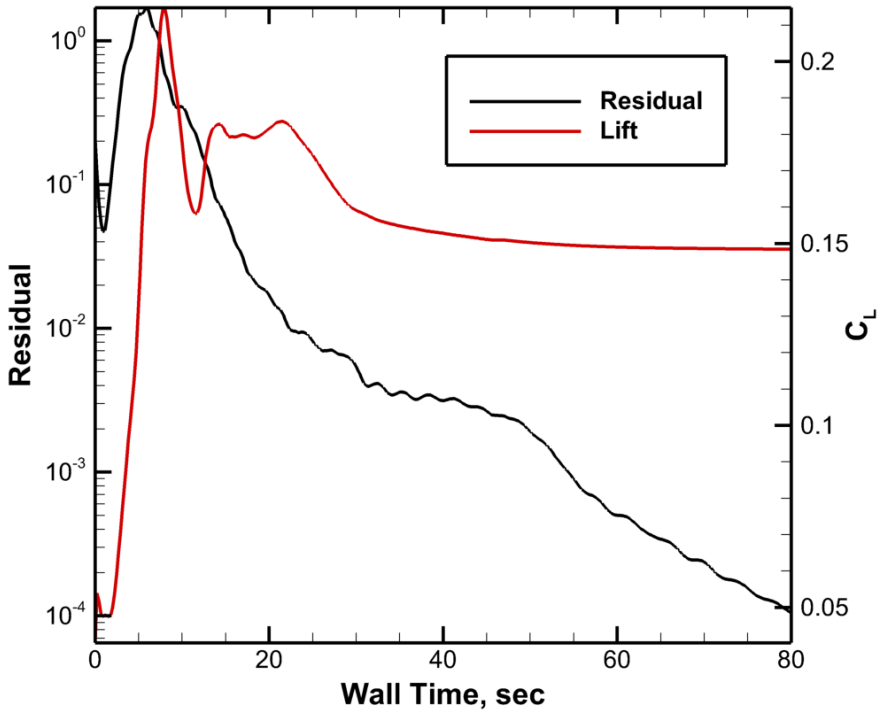
```
#PBS -S /bin/csh
#PBS -N run_test
#PBS -r n
#PBS -m ae
#PBS -M eric.j.nielsen@nasa.gov

#PBS -l select=2:ncpus=36:mpiprocs=32:model=sky_gpu:ngpus=4:mem=300g
#PBS -l place=scatter:excl
#PBS -l walltime=0:10:00
#PBS -q v100@pbspl4


module use --append /swbuild/fun3d/shared/fun3d_users/modulefiles # NASA ONLY
module purge                                                      # NASA ONLY
module load FUN3D_AVX512/13.7                                     # NASA ONLY

unsetenv CUDA_VISIBLE_DEVICES                                     # NASA ONLY

((mpiexec_mpt -np 64 nodet_mpi --time_timestep_loop ) > test.out) >& error.out
```

Running Multiple MPI Ranks per GPU

Screen Output

```
FUN3D 13.6-237971ad24 Flow started 10/12/2020 at 08:09:30 with 64 processes
Contents of fun3d.nml file below-----------------------
&project
  project_rootname = 'dpw-wbt0_med-10Mn_5.merged'
/
.
.
.
CUDA MPS status is good: either not needed or running properly on all 2 nodes.
.
.
.
   60  0.131023606510773E+00  0.14420E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       0.657988678847277E+00  0.52227E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.176608990442695E+00          Drag  0.154200246390409E-01
.22486705 seconds to complete timestep on the master rank.
   61  0.120097956449676E+00  0.11736E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       0.636071451756599E+00  0.48195E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.178735835225606E+00          Drag  0.148265470180802E-01
.14993023 seconds to complete timestep on the master rank.
   62  0.108185586213013E+00  0.85668E+02  0.16541E+05  0.00000E+00 -0.53707E+02
       0.630569363594978E+00  0.45850E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.180267154538958E+00          Drag  0.144054614883000E-01
.21819975 seconds to complete timestep on the master rank.
.
.
.
88.757 seconds to complete main timestep loop on the master rank.
Done.
```

- Running with 64 MPI ranks
- MPS is now running on all nodes
- Nominal time step costs 0.22 seconds
  - As we converge, Jacobian evaluations are frequently skipped, reducing per-step costs to 0.15 seconds

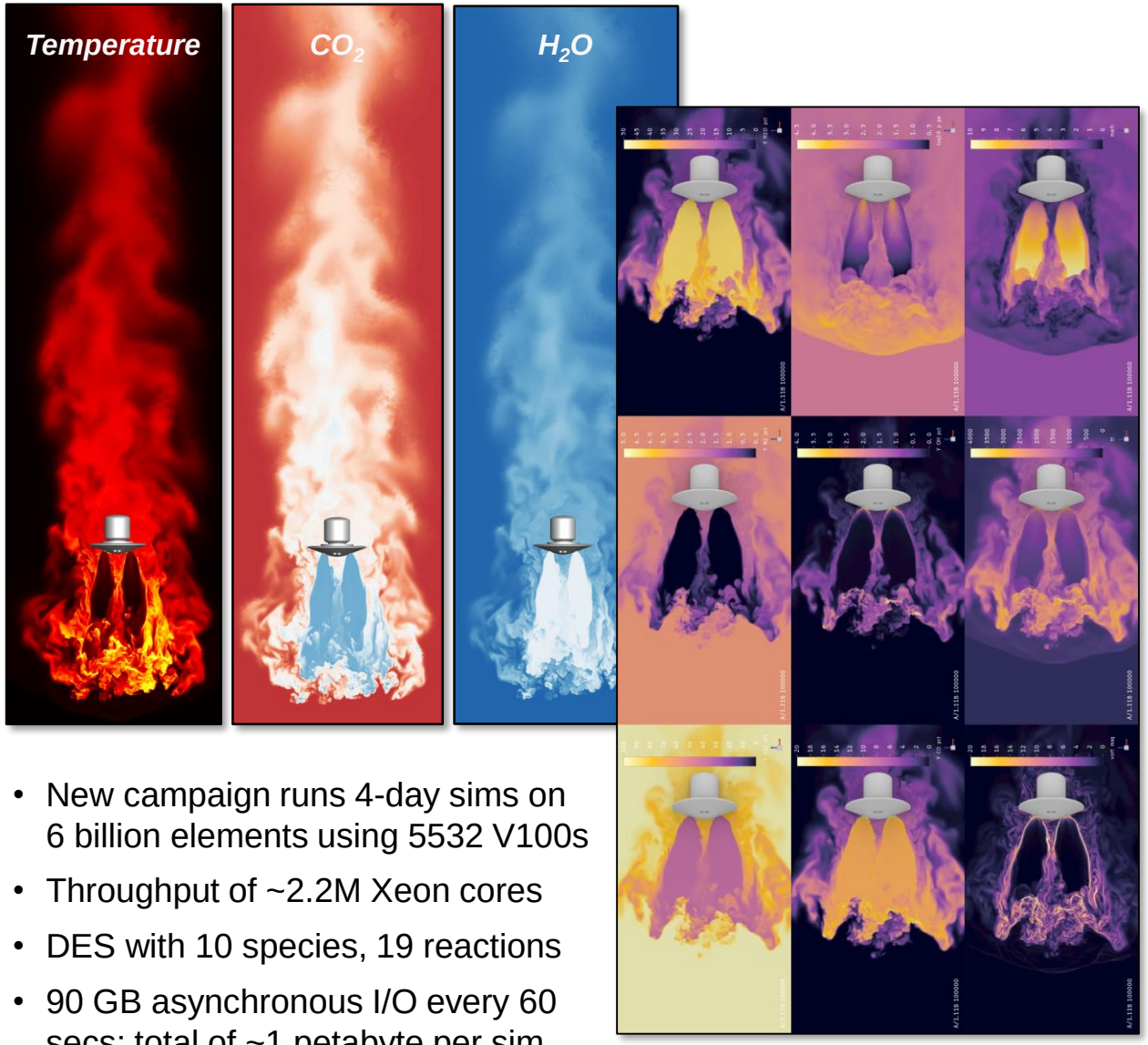


General Tips and Guidance

- For many more tips and general troubleshooting advice, see the GPU chapter of the FUN3D user manual and/or contact fun3d-support@lists.nasa.gov
- You may find that FUN3D does not function correctly at first on newly-installed GPU systems
  - We have tried to anticipate a broad range of issues we have encountered before, but please be patient: there can be many details beyond a CPU-only system
  - System administrators are sometimes unfamiliar with subtle details of GPU computing and may have set up the system in an unexpected configuration
  - Please contact fun3d-support@lists.nasa.gov for assistance
  - If we cannot help you identify/solve a problem, NVIDIA is offering tech support to the broader FUN3D community – we can connect you with the appropriate NVIDIA POC

What's Coming?

Thermochemical Nonequilibrium, Algorithmic, and Other Capabilities



- New campaign runs 4-day sims on 6 billion elements using 5532 V100s
- Throughput of ~2.2M Xeon cores
- DES with 10 species, 19 reactions
- 90 GB asynchronous I/O every 60 secs; total of ~1 petabyte per sim

NASA Ames Vis Team

Summit Performance for Unsteady 10-Species Reacting Turbulent Flow